DDPG(Continuous Control With Deep Reinforce learning) (1)

https://arxiv.org/abs/1509.02971

▼ Introduction

해당 논문은 DQN 논문을 많이 이용하였다. DQN은 이산 환경에서 저차원의 행동 공간 관측에 대해 문제를 해결할 수 있었지만, RL이 발전함에 따라서 우리는 고차원적이고 연속적인 행동공간에서 문제를 해결한다.

▼ 이산 환경에 대한 알고리즘을 연속 환경에서 적용하는 방법

논문에서는 가장 단순하게 연속 환경을 쪼개어서 이산환경으로 보고 처리하자 라는 아이디어를 제시한다. 하지만, 연속적인 공간을 여러개로 나누면 나누어진 공간수를 N이라고 하였을 때 $action^N$개의 선택지가 생긴다. 이는 지수적으로 증가하기에 더 많은 공간으로 분할함에 따라서 기하급수적으로 커지는 action수를 대면하게 된다. 따라서 모든 경우에 대해 학습하는 과정에서 차원의 저주를 피할 수 없게 된다. 이러한 이유로 이산 환경에 대한 알고리즘은 연속 환경에 적용시키기엔 한계가 존재한다.

따라서 논문에서는 **고차원적이고 연속적인 행동공간에 적용이 가능한 off-policy, actor-critic, model-free하며 심층 함수를 사용해 근사시킨 알고리즘**을 제안한다.

해당 알고리즘은 DPG(deterministic policy gredient) 기반이다.

단순히 actor-critic method를 인공 신경망 함수에 근사시키는 것은 불안정하다.
이를 완화시키는 방안이 DQN에서 제시되었다.

1. Replay Buffer를 사용하여 MDP에서 스텝간의 연관성으로 비롯되는 공분산 문제(correlation)를 최소화할 수 있었고, 이를 통해 조금은 안정적인 학습이 가능하게 하였다.
2. $y - \hat{y}$ 형태로 로스를 계산하게 되는데, 두 변수는 같은 파라미터에 대해 파라미터화 되어있다. 그리고 매 타임스텝마다 파라미터가 업데이트되게 되는데, 타겟이 계속해서 변경됨에 따라서 학습이 불안정하게 진행된다. 이를 개선하고자 back up model을 두었다. 해당 모델로 타겟을 예측함으로써 어느정도 고정된 타겟값으로 모델이 학습하게 되었고, 이를 통해 학습이 보다 안정적이게 진행되었다.

3. 부가적으로 Batch Normalization(BN)을 이용하여 모든 입력 샘플의 스케일을 조정하였고, 이를 통해 모델에 robust한 특성을 추가해주었다.

동일 파라미터값과 동일 네트워크 구조를 이용해 다양한 저차원 관측값을 이용해 정책을 학습시킬 때 DDPG(Deep DPG)는 경쟁력이 있었다.

→ 다양한 환경에 대해 동일 모델과 파라미터가 좋은 성과를 내는 모습에서 BN을 추가해 robust한 특성을 넣은게 유의미하였음을 알 수 있다.

단순 actor-critic 구조에 나머지는 수식으로 구현되었기에 확장이 용이하다는 장점이 있다.

▼ background

기본적인 강화학습은 매 타임스텝마다 환경과 소통하는 에이전트가 가정된다.

매 타임스텝 t마다 에이전트는 상태값 x_t를 받고, 그에 따라 액션 a_t을 취하며, 액선에 따른 보상 r_t를 받는다.

환경은 부분적으로 관측될 수 있기 때문에 현 상태를 표현하기 위해 현 상태까지 오게 된 궤적을 필요로 한다. 환경은 관측값과 상태가 동일한 fully-observed한 환경이기에, $s_t = x_t$이다.

이에 따라서 궤적은 $s_t = (x_1, a_1, ..., a_{t-1}, x_t) = (s_1, a_1, ..., a_{t-1}, s_t)$로 표현된다.

▼ fully-observed

agent의 action은 env의 observation에 의해 결정된다. agent가 모든 state를 observation할 수 있을 때를 fully-observed라고 표현한다.

반대되는 표현으로는 partially-observed가 있다.

에이전트의 행동은 정책 π에 의해 결정된다.

감가율 $\gamma = [0, 1]$일 때 $R_t = \sum_{i=t}^{T} \gamma^{(i-t)} r(s_i, a_i)$이다.

ρ^{π}는 정책 π에 대한 discounted state visitation distribution이다.

$$Q^{\pi}(s_t, a_t) = E_{r_t, s_{t+1}}[r(s_t, a_t) + \gamma E_{a_{t+1}}[Q^{\pi}(s_{t+1}, a_{t+1})]]$$

벨만 방정식에 따른 Q 함수의 재귀적 구조

$$Q^{\mu}(s_t, a_t) = E_{r_t, s_{t+1}}[r(s_t, a_t) + \gamma Q^{\mu}(s_{t+1}, \mu(s_{t+1}))]$$

앞선 벨만방정식에서 stochastic하게 action을 뽑아내던 것을 deterministic하게 action을 고름으로써 action에 대한 expectation term이 사라졌다.

이를 actor-critic에 알고리즘에 적용하게 되면

$$L(\theta^Q) = \mathbb{E}_{s_t \sim \rho^\beta, a_t \sim \beta, r_t \sim E}\left[\left(Q(s_t, a_t|\theta^Q) - y_t\right)^2\right]$$

$$y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, \mu(s_{t+1})|\theta^Q).$$

이 된다.

▼ Algorithm

Q learning은 모든 action space에 대해 탐색하며 결과적으로는 가장 좋은 길을 greedy하게 고른다. 이는 시간만 충분하다면 성능이 보장되는 방법인데, 앞선 introduction에서 말하였듯 행동공간의 선택폭이 기하급수적으로 늘어남에 따라서 다음과 같은 방식은 제한적이다.

state와 action를 매핑시킨 μ라는 함수를 만들어냄

$$\begin{aligned}\nabla_{\theta^\mu} J &\approx \mathbb{E}_{s_t \sim \rho^\beta}\left[\nabla_{\theta^\mu} Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t|\theta^\mu)}\right] \\ &= \mathbb{E}_{s_t \sim \rho^\beta}\left[\nabla_a Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta_\mu} \mu(s|\theta^\mu)|_{s=s_t}\right]\end{aligned}$$

다음과 같은 수식으로 업데이트

NFQCA는 batch learning을 통해 안정성을 올리고 있는데, stochastic GD와 batch GD의 차이와 유사했다.

인공지능을 이용한 강화학습은 I.I.D를 추정하려고 한다. 하지만 강화학습은 MDP 특성상 공선성문제가 발생할 수 밖에 없는 구조이다.

DQN에서 유한한 크기의 캐시 메모리 R에 (s_t, a_t, r_t, s_{t+1})을 저장하고, 용량이 다 차면 오래된 정보부터 삭제하게 하였다. replay buffer의 사이즈가 커질수록 공선성은 더욱 더 줄어들게 된다. 또한 업데이트시 미니배치는 균일한 확률분포에서 샘플링된다. NFQCA의 batch learning의 장점도 가지고 온 모습이다.

백업 모델에 업데이트를 해줄 때 일정 비율(τ)만큼 파라미터를 넘겨줌으로써 급격한 변화로 모델의 수렴이 흔들리는 것을 완화하였다.

다양한 환경에서 robust하게 작용하게 하려고, BN을 입력전, 각 모델의 레이어마다 실시해 주었다.

DPG는 분포에서 action을 sampling하던 Stochastic과는 달리 action을 바로 골라낸다. 따라서 탐험을 하게 하는 방법으로 noise를 넣어주게 되었다. 논문에서는 OU noise를 사용하였다.

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for